Claims:

1. A method creating an augmented Mealy machine from a Mealy machine having an input space Σ, a state space Q and an output space Δ, the method comprising the steps of:

A. selecting as a state q_a one of (A1) a dedicated output stopping state from the state space θ that is not reachable from any other state in Q such that $Q' = Q$ and (A2) an additional stopping state not in θ if no dedicated output stopping state exists in Q such that $Q' = Q \cup \{ q_a \}$;

B. setting $\Delta' = \Delta$ and adding a dedicated blank symbol B to Δ' only if Δ does not already contain a dedicated blank symbol B;

C. modifying the state-transition mapping δ to δ' such that the augmented Mealy machine always remains in q_a once q_a is entered;

D. modifying the output mapping λ to λ' such that the augmented Mealy machine always outputs B once q_a is entered; and

E. modifying a domain D of the state-transition mapping.

2. The method as claimed in claim 1, further comprising the steps of:

F. selecting an integer N' based on the size of the state, input, and an output spaces;

G. selecting a number N, no less than N'; and

H. determining a vectorization of input, state, and output spaces resulting in vectors over Z_N, wherein N is prime if the Mealy machine is represented using polynomials.

3. The method as claimed in claim 2, further comprising the steps of:

F. adding dummy states until Q' contains a number of states determined by a vectorization of the state space using N;

G. adding dummy input symbols until Σ' contains a number of symbols determined by the vectorization of the input space using N;

H. adding dummy output symbols until Δ' contains a number of symbols determined by the vectorization of the output space using N;

I. for each pair $(q,\sigma) \notin D'$, setting $\delta'(q,\sigma) = q_a$ and $\lambda'(q,\sigma) = B$.

4. The method as claimed in claim 2, further comprising the steps of:

F. adding states until Q' contains a number of states determined by the vectorization of the state space using N according to the sub-steps of:

i. selecting a random $q \in Q$ (alternatively in the current $Q' - \{q_a\}$),

ii. adding a state q' to Q',

iii. for every $\sigma\in\Sigma$ setting $\delta'(q',\sigma)=\delta'(q,\sigma)$ and $\lambda'(q',\sigma)=\lambda'(q,\sigma)$,

iv. optionally also for every pair $(q,\sigma)\in Q'\times\Sigma$ such that $\delta'(q,\sigma)=q$, randomly setting $\delta'(q,\sigma)$ to be one of q and q' and randomly setting $\delta'(q',\sigma)$ to be one of q and q';

G. modifying D to D' to reflect the addition of the new definitions;

H. adding dummy input symbols to Σ' until it contains a number of symbols determined by the vectorization of the input space using N;

I. adding dummy output symbols to Δ' until it contains a number of symbols determined by the vectorization of the output space using N;

J. for each pair $(q,\sigma)\notin D'$, setting $\delta'(q,\sigma)$ to a random $q'\in Q'$ and setting $\lambda'(q,\sigma)$ to a random symbol from Δ'.

5. The method as claimed in claim 2, further comprising the steps of:

F. adding dummy states until Q' contains a number of states determined by the vectorization of the state space using N;

G. adding dummy input symbols until Σ' contains a number of symbols determined by the vectorization of the input space using N;

H. adding dummy output symbols until Δ' contains a number of symbols determined by the vectorization of the output space using N;

I. for each pair $(q,\sigma)\in D'$, setting $\delta'(q,\sigma)$ to a random $q'\in Q'$, setting $\lambda'(q,\sigma)$ to a random output symbol in Δ'.

6. The method as claimed in claim 1, further comprising the step of:

F. interchanging states at random, and adjusting δ' and λ' accordingly;

G. interchanging symbols in the extended input alphabet Σ', and adjusting δ' and λ' accordingly; and

H. interchanging symbols in the extended output alphabet Δ', and adjusting δ' and λ' accordingly.

7. A method of transforming state-transition and output mappings of an augmented Mealy machine to polynomial mappings, the method comprising the steps of:

computing interpolations of δ' and λ' from the state-transition and output mappings; and

performing polynomial interpolation of the original state-transition and output mappings.

8. The method as claimed in claim 7, further comprising the step of pre-computing coefficients of $a_i(x)$ functions, wherein the step of performing polynomial interpolation of the original mappings uses the pre-computed coefficients of the $a_i(x)$ functions.

9. A method of converting a Blum-Shub-Smale machine to a BSS' machine, comprising:

A. selecting an integer N such that: (1) N is at least as large as the number of nodes, (2) N is greater than user-specified constants used in defining the Blum-Shub-Smale machine, (3) N is a prime number;

B. restricting a ring R to only integer fields of a form $\mathbb{Z}_N$;

C. restricting computation mappings to only polynomial mappings;

D. providing a new node-numbering convention;

E. changing comparisons in branch nodes to set membership relations of the form $\in K$, where $K \subseteq (\mathbb{Z}_N - \{0\})$;

F. revising a definition of a full state space to include a current node number, an internal state space, and output and input spaces to produce a revised full state space;

G. restricting computation mappings, g_n, to an identity mapping for input components of a revised full state vector;

H. restricting all computation mappings, g_n, such that no output components in the full state vector may be used in further computation;

I. requiring that at least one computation mapping uses at least one input vector component;

J. requiring that all halting nodes generate output; and

K. restricting node to (1) computation nodes, which may contain at least one of computations and branches, and (2) halting nodes.

10. A method of specifying a BSS' machine, comprising the steps of:

A. specifying a set of nodes;

B. specifying computation mappings for each node;

C. specifying a next-node function β, along with set membership relations;

D. selecting an integer N such that: (1) N is at least as great as a number of nodes, (2) N is greater than user-specified constants used in defining a corresponding Blum-Shub-Smale machine, and (3) N is a prime number; and

E. specifying a vectorization of the state, input, and output spaces.

11. The method as claimed in claim 9, further comprising at least one of the steps of:

listing explicity the halting nodes of the BSS' machine; and
explicitly specifying an output symbol B as a halting signal

12. The method as claimed in claim 10, further comprising at least one of the steps of:
listing explicity the halting nodes of the BSS' machine; and
explicitly specifying an output symbol B as a halting signal.

13. A method of transforming a BSS' machine to a multivariate polynomial mapping, comprising the steps of:

A. acquiring a specification of a BSS' machine;

B. expressing a set membership relation $\in K$, $K \subseteq (\mathbb{Z}_N - \{0\})$, as a polynomial;

C. expressing a next-node function β as a polynomial; and

D. expressing a complete computing endomorphism H as a multivariate polynomial mapping.

14. A method of transforming a BSS' machine to a mapping represented as a function table, comprising the steps of:

A. selecting a specification of a BSS' machine if no single multivariate polynomial computing endomorphism H is given;

B. selecting a multivariate polynomial mapping H for the BSS' machine;

C. converting state and input vectors, both with base-N components, to a base N^{1+S+I} number X, where $1+S+I$ is a total number of components in the state and input vectors, wherein the state vector includes a node number;

D. converting the output of the computing endomorphism H with base-N components, to a base N^{1+S+O} number F, where $1+S+O$ is the total number of components in the output of H;

E. inserting F into the Xth entry in the function table; and

F. repeating steps (C)-(E) for all possible state vector/input vector combinations.

15. A method of computing with a BSS' machine transformed to a multivariate mapping comprising the steps of:

A. initializing the BSS' machine;

B. applying the multivariate mapping to the full state space vector;

C. checking if the machine has halted, and ending computation if the machine has halted; and

D. repeating steps (B) and (C) if the machine has not halted.

16. The method as claimed in claim 15, wherein the step of applying further comprises at least one of (1) reading the output vector produced by the machine and (2) changing the input vector of the machine.

17. The method as claimed in claim 15, wherein the step of initializing comprises the sub-steps of (1) specifying a node number; (2) specifying a remaining initial state vector; and (3) specifying an initial input vector.

18. A method of specifying a pattern of encryption of multivariate mappings with univariate mappings comprising the steps of:

A. specifying a number d of variables of the multivariate mapping;

B. specifying a number e of mapping components of the multivariate mapping;

C. selecting variables to be used in encrypted form;

D. selecting mapping components to be encrypted; and

E. selecting equality restrictions to be placed on components in the key pairs.

19. A method of generating keys for univariate encryption of multivariate mappings, comprising the steps of:

A. determining a key representation;

B. acquiring a specified pattern of encryption;

C. determining a number of key pairs to be generated from an acquired pattern of encryption; and

D. permuting elements of field $\mathbb{Z}_N$ while simultaneously recording data for the permutation and its inverse in two arrays R and S once for each unique key pair generated.

20. The method as claimed in claim 19, wherein the multivariate mappings to be encrypted are expressed using polynomials, further comprising the step of computing the permutation and its inverse by interpolation, using $a_i(x)$; once for each unique key pair that is to be generated.

21. The method as claimed in claim 19, wherein the multivariate mappings to be encrypted are expressed using polynomials, further comprising the steps of:

precomputing arithmetic operations over the field $\mathbb{Z}_N$; and precomputing coefficients of functions $a_i(x)$, wherein the step of computing the permutation and its inverse by interpolation uses the precomputed coefficients of functions $a_i(x)$ and the precomputed arithmetic operations.

22. The method as claimed in claim 19, wherein the steps of determining a number of key pairs further comprises the step of restricting possible keys according to a user input.

23. The method as claimed in claim 19, further comprising the step of setting keys in the key pairs, that are neither to encrypt nor decrypt, to the identity mapping.

24. A method of encrypting plural variables and mapping components of multivariate mappings, represented, with univariate mappings of an appropriate representation, comprising the steps of:

A. determining a representation for the encryption;

B. replacing each variable to be decrypted x_i with a decrypted equivalent $s_{e+i}(x_i)$;

C. composing the decrypted equivalents with a mapping h; and

D. composing each mapping component to be encrypted, h_i, with an encryption function r_i, to create $r_i(h_i(\ldots))$.

25. The method as claimed in claim 24, wherein the multivariate mappings comprise mappings represented as one of function tables and polynomials.

26. A method of generating re-encryption keys for re-encryption of plural variables and mapping components of multivariate mappings, already partially encrypted with first univariate functions, with second univariate functions, comprising the steps of:

A. determining a representation for the re-encryption;

B. acquiring key pairs corresponding to the first univariate functions;

C. generating a new set of key pairs $(r'_1, s'_1), \ldots, (r'_{n+m}, s'_{n+m})$;

D. symbolically composing r'_i with s_i for $1 \le i \le n$; and

E. symbolically composing r_i with s'_i for $n < i \le n+m$.

27. The method as claimed in claim 26, further comprising the step of acquiring a pattern of encryption used for the first univariate functions, wherein the step of generating comprises generating the new set of key pairs based on the acquired pattern.

28. A method of re-encryption of plural variables and mapping components of multivariate mappings already partially encrypted with first univariate functions with second univariate functions, comprising the steps of:

A. determine a representation for re-encryption;

B. symbolically substituting the i^{th} variable x_{i-n} with $r_i(s_i'(x_{i-n}))$ for all $n<i\leq n+m$; and

C. symbolically composing $r_i'(s_i(x))$ with the i^{th} function component f_i for all $1\leq i\leq n$.

29. The method as claimed in claim 28, further comprising the step of acquiring a corresponding set of pairs of re-encryption keys, wherein the step symbolically composing comprises composing using the acquired keys.

30. A method of converting a mapping $t: Z_N^m \to Z_N^n$, given as a function table, into a function $t': Z_{N^m} \to Z_{N^n}$, also given as a function table, comprising the steps of:

A. computing $X=N^{m-1}x_m+\ldots+N^1x_2+x_1$ for a vector $(x_1,\ldots,x_m)\in Z_N^m$;

B. computing $F=N^{n-1}f_n+\ldots+N^1f_2+f_1$ for an entry $(f_1,\ldots,f_n)$ corresponding to $(x_1,\ldots,x_m)$;

C. setting $t'(X)=F$; and

D. repeating for all $(x_1,\ldots,x_m)\in Z_N^m$.

31. A method of converting a function $t: Z_{N^m} \to Z_{N^n}$, given as a function table, into a function $t': Z_N^m \to Z_N^n$, also given as a function table, comprising the steps of:

A. computing $X=N^{m-1}x_m+\ldots+N^1x_2+x_1$

B. reducing $t(X)$ to a base-N representation, such that $t(X)$ is represented by a tuple $(f_1,\ldots,f_n)$;

C. setting the tuple in t', indexed by $(x_1,\ldots,x_m)$ to $(f_1,\ldots,f_n)$; and

D. repeating for all $(x_1,\ldots,x_m)\in Z_N^m$.

32. A method of symbolically composing mappings $f: Z_N^m \to Z_N^n$ and $g: Z_N^n \to Z_N^o$ represented as function tables to produce the function table for $g(f(x))$, comprising the steps of:

A. generating a new function $t_f: Z_{N^m} \to Z_{N^n}$ represented by a function table, where every mapping value $(f_1,\ldots,f_n)$ corresponding to a $(x_1,\ldots,x_m)$ is placed in entry number $X=N^{m-1}x_m+\ldots+N^1x_2+x_1$ as the number $F=N^{n-1}f_n+\ldots+N^1f_2+f_1$;

B. generating a new function $t_g: Z_{N^m} \to Z_{N^n}$ represented by a function table, where every mapping value $(f_1,\ldots,f_n)$ corresponding to a $(x_1,\ldots,x_m)$ is placed in entry number $X=N^{m-1}x_m+\ldots+N^1x_2+x_1$ as the number $F=N^{n-1}f_n+\ldots+N^1f_2+f_1$; and

C. for each X from 0 to N^m-1 setting $t_{gf}(X)=t_g(t_f(X))$, where t_{gf} is a function table for a function $t_{gf}: Z_{N^m}\to Z_{N^n}$.

33. A method of specifying a pattern of encryption of multivariate mappings with other multivariate mappings comprising the steps of:

A. selecting a number of mapping components c_i to be grouped together, $c_i\geq 1$, for in all l successive groups of components, covering all components in a mapping only once;

B. selecting a number of variables c_i to be grouped together, $c_i\geq 1$, for in all k - l successive groups of variables, covering all variables in a mapping only once;

C. selecting groups of components to be encrypted;

D. selecting groups of variables to be used in encrypted form; and

E. selecting equality restrictions to be placed on components in the key triples.

34. A method of generating keys for multivariate encryption of multivariate mappings, comprising the steps of:

A. determining a representation for the keys;

B. defining for an i^{th} key triple two temporary $N^{c_i}\times(c_i+1)$ arrays R and S;

C. defining a temporary polynomial f to translate from base-N vectors to base-N^{c_i} vectors with c_i components; and

D. permuting a ring $Z_{N^{c_i}}$, and simultaneously translating a permutation and its inverse to a field $Z_N^{c_i}$ for every key triple;

E. repeating steps B-D for all triples not set equal to identity.

35. The method as claimed in claim 34, wherein the mappings to be encrypted are expressed using polynomials, further comprising the step of computing the permutation and its inverse by interpolation, using at least a portion of R and S as interpolation data, using $a_i(x)$, once for each unique key triple that is to be generated.

36. The method as claimed in claim 34, further comprising the step of setting all key triples that are to do neither encryption nor decryption to the identity mapping.

37. The method as claimed in claim 34, further comprising the steps of (1) pre-computing arithmetic operations over the field Z_N and (2) pre-computing coefficients of the functions $a_j(x)$, wherein the steps of permuting comprises using the pre-computed $a_j(x)$.

38. The method as claimed in claim 34, further comprising the step of restricting a new set of key triples based on a pattern of encryption used during an encryption of the first multivariate polynomials.

39. A method of encrypting plural groups of variables and groups of mapping components of multivariate mappings, with other multivariate mappings, comprising the steps of:

A. determining a mapping representation for encryption;

B. replacing each group of encrypted variables $\vec{w}_i$ with a decrypted equivalent $s_{i+l}(\vec{w}_i)$;

C. composing each of the decrypted equivalents with a mapping h; and

D. composing each group of mapping components to be encrypted v_i with r_i giving $r_i(v_i(\ldots))$.

40. A method of generating re-encryption keys for re-encryption of plural variables and mapping components of multivariate mappings already partially encrypted with first multivariate mappings with second multivariate mappings comprising the steps of:

A. determining a representation for keys;

B. acquiring key triples used with the first multivariate mappings;

C. generating a new set of key triples $(c_1', r_1', s_1'), \ldots, (c_k', r_k', s_k')$;

D. symbolically composing r_i' with s_i for $1 \le i \le l$; and

E. symbolically composing r_i with s_i' for $l < i \le k$.

41. The method as claimed in claim 40, wherein the multivariate mappings comprise one of polynomials and function tables.

42. A method of re-encrypting plural variables and mapping components of multivariate mappings already partially encrypted with first multivariate mappings with second multivariate mappings, the method comprising the steps of:

A. determining a representation for the re-encryption;

B. acquiring key triples used for the encryption using the first multivariate mappings;

C. symbolically substituting an $i-l^{th}$ variable block $\vec{w}_{i-l}$ with $r_i(s_i'(\vec{w}_{i-l}))$, for all $l < i \le k$; and

D. symbolically composing $r_i'(s_i(\vec{w}))$ with an i^{th} function component block $\vec{v}_i$ for all $1 \le i \le l$.

43. A method of symbolically composing mappings $f{:}Z_N^m \to Z_N^n$ and $h_1, \ldots, h_k{:}Z_N^{c_i} \to Z_N^{c_i}$, represented as function tables, to produce the function table for $f(h_1(x_1, \ldots, x_{c_1}), h_2(x_{c_1+1}, \ldots, x_{c_1+c_2}), \ldots, h_k(x_{m-c_k+1}, \ldots, x_m))$, comprising the steps of:

A. acquiring a specification for the c_i such that $\sum_{i=1}^{k} c_i=m$;

B. generating a new function $t_f{:}Z_{N^m}\to Z_{N^n}$ represented by a function table, where every mapping value $(f_1,...,f_n)$ corresponding to a $(x_1,...,x_m)$ is placed in entry number $X=N^{m-1}x_m+...+N^1x_2+x_1$ as the number $F=N^{n-1}f_n+...+N^1f_2+f_1$;

C. for each i generating a new function $t_{h,i}{:}Z_{N^{c_i}}\to Z_{N^{c_i}}$ represented by a function table where every mapping value $(h_{i,1},...,h_{i,c_i})$ corresponding to a $(x_{a+1},...,x_{a+c_i})$, where $a=\sum_{j=1}^{i-1} c_i$, is placed in entry number $X=N^{c_i-1}x_{a+c_i}+...+N^1x_{a+2}+x_{a+1}$ as the number $H=N^{c_i-1}h_{i,c_i}+...+N^1h_{i,2}+h_{i,1}$.

D. initializing a new function table for a function $t_{fh}{:}Z_{N^m}\to Z_{N^m}$;

E. for every i from 1 to k setting $y_i=N^{c_i}$;

F. for every i from 0 to N^m-1:

```
i.    setting $u=0$;
ii.   for every $j$ from $k$ to 1 setting $u=y_ju+t_{fh}(b_j)$;
iii.  setting $t_{fh}(i)=t_f(u)$; and
iv.   incrementing the vectorized index $(b_1,...,b_k)$, taking into account that $b_1$ is
      in base $y_1$, $b_2$ is in base $y_2$, ..., $b_k$ is in base $y_k$.
```

44. A method of symbolically composing mappings $f{:}Z_N^m\to Z_N^n$ and $h_1,...,h_k{:}Z_N^{c_i}\to Z_N^{c_i}$ represented as function tables to produce the function table for $(h_1(f_1(x_1,...,x_m),...,f_{c_1}(x_1,...,x_m)),...,h_k(f_{n-c_k+1}(x_1,...,x_m),...,f_n(x_1,...,x_m)))$, comprising the steps of:

A. acquiring a specification for the c_i such that $\sum_{i=1}^{k} c_i=n$;

B. generating a new function $t_f{:}Z_{N^m}\to Z_{N^n}$ represented by a function table, where every mapping value $(f_1,...,f_n)$ corresponding to a $(x_1,...,x_m)$ is placed in entry number $X=N^{m-1}x_m+...+N^1x_2+x_1$ as the number $F=N^{n-1}f_n+...+N^1f_2+f_1$;

C. for each i generating a new function $t_{h,i}{:}Z_{N^{c_i}}\to Z_{N^{c_i}}$ represented by a function table where every mapping value $(h_{i,1},...,h_{i,c_i})$ corresponding to a $(f_{a+1},...,f_{a+c_i})$, where $a=\sum_{j=1}^{i-1} c_i$, is placed in entry number $X=N^{c_i-1}f_{a+c_i}+...+N^1f_{a+2}+f_{a+1}$ as the number $H=N^{c_i-1}h_{i,c_i}+...+N^1h_{i,2}+h_{i,1}$.

D. initializing a new function table for a function $t_{hf}{:}Z_{N^m}\to Z_{N^m}$;

E. setting $y_0=1$, $y_1=1$;

F. for every i from 2 to k setting $y_i=y_{i-1}N^{c_{i-1}}$;

G. initializing a vector $(b_1,...,b_k)$ to $(0,...,0)$;

H. for every i from 0 to N^n-1:

```
i.    setting $u=t_f(i)$;
ii.   setting $q=0$;
iii.  for every $j$ from $k$ to 1:
```

a. setting p to the integer result of u/y_j;

b. setting $u = u - py_j$ to get the remainder;

c. setting p to the integer result of u/y_{j-1};

d. setting $b_j = t_{h,j}(p)$;

e. adding $y_j b_j$ to q; and

iv. setting $t_{hj}(i) = q$.

45. A Turing platform, supporting unencrypted and partially encrypted polynomial computation for some machine *M*, on a host O, comprising:

A. a Turing machine;

B. a register writeable by a finite control of the Turing machine and readable by *M*;

C. a register readable by the finite control of the Turing machine and writeable by *M*;

D. a register writeable by the host $\mathbb{O}$; and

E. a register readable by the host $\mathbb{O}$.

46. A method of computing with host $\mathbb{O}$ running a Turing platform *T* supporting at least one of a Mealy or BSS' machine *M*, comprising:

A. initializing the machine;

B. initialize the Turing platform;

C. reading, by *T*, a storage cell at which its finite control is located;

D. writing, by *T*, a contents of the storage cell to a register readable by *M*;

E. writing, by $\mathbb{O}$, to a remaining input of *M*;

F. executing, by $\mathbb{O}$, one computation step for *M*;

G. writing, by $\mathbb{O}$, some output of *M* to part of the register readable by *T*;

H. writing, by $\mathbb{O}$, *M*'s computed direction of movement for the finite control of *T* to a remainder of register readable by *T*;

I. writing, by $\mathbb{O}$, a rest of the output of *M* to the host O;

J. reading, by *T*, from the readable register of *T*;

K. writing, by *T*, a new cell value to storage;

L. moving the finite control of *T* one of left, right, and not at all;

M. checking by $\mathbb{O}$ to see if a halting condition has occurred; and

N. repeating steps (C)-(M).

47. The apparatus of a register machine, comprising:

A. a set $P=\left\{\vec{P}_{(i_1,\ldots,i_d)}\right\}$ of vectors of integers in Z_N^d indexed by vectors also in Z_N^d;

B. either a vector S indicating the end of the program in P, or a constant T, which functions as an instruction indicating that the computation is finished;

C. a vector $\vec{C}\in Z_N^d$ functioning as instruction pointer;

D. a set $S=\left\{\vec{S}_{(i_1,\ldots,i_d)}\right\}$ of vectors of integers in Z_N^d indexed by vectors also in Z_N^d;

E. a vector $\vec{D}\in Z_N^d$ functioning as storage pointer;

F. at least one register $(\vec{R}_1,\ldots,\vec{R}_m)$ of vectors of integers in Z_N^d for $0<m\leq N$;

G. a next instruction pointer mapping $f\left(\vec{R}_1,\ldots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right):Z_N^{d(m+4)}\rightarrow Z_N^d$;

H. a next storage pointer mapping $g\left(\vec{R}_1,\ldots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right):Z_N^{d(m+4)}\rightarrow Z_N^d$;

I. a specification of the registers that accept input from the host platform;

J. a number k of registers not accepting input from the host platform, such that $0\leq k\leq m$;

K. a register transition mapping $h\left(\vec{R}_1,\ldots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right):Z_N^{d(m+4)}\rightarrow Z_N^{kd}$;

L. a storage transition mapping $q\left(\vec{R}_1,\ldots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right):Z_N^{d(m+4)}\rightarrow Z_N^d$;

M. a specification for initializing P and S; and

N. a specification for where input is entered, including:

- through at least one register,
- through at least one storage space S,
- through an initial contents of an instruction space P, or
- through an initial contents of the at least storage space S.

48. A method of initializing a register machine comprising the steps of:

A. specifying initial values of $\vec{R}_1,\ldots,\vec{R}_m,\vec{C},\vec{D}$;

B. optionally specifying values for at least one storage cell $\vec{S}_{\vec{D}}$ in S;

C. specifying elements in P; and

D. computing values of $\vec{P}_{\vec{C}}$ and $\vec{S}_{\vec{D}}$.

49. A method of using a register machine, comprising the steps of:

A. initializing the register machine;

B. computing a next instruction pointer: $\vec{C}'=f\left(\vec{R}_1,\ldots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right)$;

C. computing a next storage pointer: $\vec{D}'=g\left(\vec{R}_1,\ldots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right)$;

D. computing a value to be written to a current storage cell: $\vec{S}'=q\left(\vec{R}_1,\ldots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right)$;

E. computing the register transition mapping: $\left(\vec{R}_{j_1},\ldots,\vec{R}_{j_k}\right)=h\left(\vec{R}_1,\ldots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right)$, where $j_1,\ldots,j_k$ specify the registers which the register machine may change;

F. setting $\vec{S}_{\vec{D}}=\vec{S}'$, $\vec{C}=\vec{C}'$, and $\vec{D}=\vec{D}'$;

G. computing $\vec{P}_{\vec{C}}$ and $\vec{S}_{\vec{D}}$ using input from a host; and

H. repeating steps (B)-(G) until a halting condition is satisfied.

50. The method as claimed in claim 48, further comprising:

I. specifying a register dedicated to output of movement direction, using an integer y such that $0<y\leq m$, and an integer z such that $0<z\leq d$;

J. specifying a register dedicated as output to a Turing platform; and

K. specifying a register dedicated as input from a Turing platform, wherein each storage unit is a vector in Z_N^d.

51. A method of using a register machine M on a host $\mathbb{O}$ comprising the steps of:

A. initializing a register machine;

B. initializing a Turing platform;

C. reading, by T, a storage cell at which its finite control is located;

D. writing, by T, a contents of the storage cell to a specified register of M;

E. writing, by $\mathbb{O}$, to a remaining specified input registers of M;

F. computing the next instruction pointer: $\vec{C}'=f\left(\vec{R}_1,\dots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right)$;

G. computing the next storage pointer: $\vec{D}'=g\left(\vec{R}_1,\dots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right)$;

H. computing the value to be written to the current storage cell: $\vec{S}'=q\left(\vec{R}_1,\dots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right)$;

I. computing the register transition mapping: $\left(\vec{R}_{j_1},\dots,\vec{R}_{j_k}\right)=h\left(\vec{R}_1,\dots,\vec{R}_m,\vec{P}_{\vec{C}},\vec{S}_{\vec{D}},\vec{C},\vec{D}\right)$, where $j_1,\dots,j_k$ specify the registers which the register machine may change;

J. setting $\vec{S}_{\vec{D}}=\vec{S}'$, $\vec{C}=\vec{C}'$, and $\vec{D}=\vec{D}'$;

K. computing $\vec{P}_{\vec{C}}$ and $\vec{S}_{\vec{D}}$;

L. reading, by T, from the readable register of M;

M. writing, by T, a new cell value to storage;

N. reading, by T, from a second readable register of M its direction of movement;

O. moving the finite control of T one of left, right, and not at all;

P. checking by $\mathbb{O}$ to see if a halting condition has occurred; and

Q. repeating steps (C)-(P).

52. A method of symbolically composing $f{:}Z_N^m\to Z_N^n$ with $h_1,\dots,h_k{:}Z_N^{d_i}\to Z_N^{c_i}$, producing the mapping $f(h_1(x_{e(1,1)},\dots,x_{e(1,d_1)}),\dots,h_k(x_{e(k,1)},\dots,x_{e(k,d_k)}))$, $\sum_{i=1}^{k} c_i=m$, comprising the steps of:

A. acquiring a specification of index selections $e(1,1),\ldots,e(1,d_1),\ldots,e(k,1),\ldots,e(k,d_k)$ deciding how variables are used in the composition;

B. initializing a vector $(a_1,\ldots,a_m)$ to $(0,\ldots,0)$;

C. performing, for every i from 0 to N^m-1

```
        1. For every $j$ from $k$ to 1 compute $h_j(a_{e(j,1)},\ldots,a_{e(j,d_j)})$ do:
                a. setting $t_{fh}(a_1,\ldots,a_m)=f(h_1,\ldots,h_k)$;
                b. increment a vectorized index $(a_1,\ldots,a_m)$.
```

53. A method of symbolically composing $h_1,\ldots,h_k$ with f producing the mapping

$$(h_1(f_{e'(1,1)}(x_1,\ldots,x_m),\ldots,f_{e'(1,c_1)}(x_1,\ldots,x_m),x_{e(1,1)},\ldots,x_{e(1,d_1)}),\ldots,$$
$$h_k(f_{e'(k,1)}(x_1,\ldots,x_m),\ldots,f_{e'(k,c_k)}(x_1,\ldots,x_m),x_{e(k,1)},\ldots,x_{e(k,d_k)})),$$

comprising the steps of:

A. acquiring a specification of index selections $e'(1,1),\ldots,e'(1,c_1),\ldots,e'(k,1),\ldots,e'(k,c_k)$ deciding how mapping components are used in the composition;

B. acquiring a specification of index selections $e(1,1),\ldots,e(1,d_1),\ldots,e(k,1),\ldots,e(k,d_k)$ deciding how variables are used;

C. setting a vector $(a_1,\ldots,a_m)$ to $(0,\ldots,0)$.

D. For every i from 0 to N^m-1 do:

```
        1. For every $j$ from $k$ to 1 compute $h_j(f_{e'(j,1)},\ldots,f_{e'(j,c_j)},x_{e(j,1)},\ldots,x_{e(j,d_j)})$;
                a. Setting $t_{hf}(a_1,\ldots,a_m)=(h_1,\ldots,h_k)$.
                b. Increment the vectorized index $(a_1,\ldots,a_m)$.
```

54. A method of specifying a pattern of parametrized encryption of multivariate mappings with other multivariate mappings comprising the steps of:

A. selecting a number of mapping components c_i to be grouped together, $c_i \geq 1$, for all l successive groups of components, covering all components in a mapping only once;

B. selecting a number of variables c_i to be grouped together, $c_i \geq 1$, for all $k - l$ successive groups of variables, covering all variables in a mapping only once;

C. selecting a group of variables g_i, $l < g_i \leq k$, to be used as "parameter", or explicitly selecting no such "parameter" for all k successive groups of components and variables;

D. selecting groups of mapping components to be encrypted;

E. selecting groups of variables to be used in encrypted form; and

F. selecting equality restrictions to be placed on components in the key quadruples.

09723742.112800

55. A method of generating keys for parametrized multivariate encryption of multivariate mappings, comprising the steps of:

A. determining an appropriate representation for the keys;

B. defining for an i^{th} key quadruple that is to do parametrized encryption/decryption two temporary $N^{c_i+c_{g_i}}\times(c_i+1)$ arrays R and S;

C. defining for an i^{th} key quadruple that is to do non-parametrized encryption/decryption two temporary $N^{c_i}\times(c_i+1)$ arrays R and S;

D. repeating steps (A)-(C) for all i quadruples;

E. defining a temporary polynomial f to translate from base-N vectors to base-N^{c_i} with c_i components;

F. permuting a ring $Z_{N^{c_i}}$, and simultaneously translating the permutation and its inverse to the field $Z_N^{c_i}$;

G. repeating step (F) in all $N^{c_{g_i}}$ times while recording data, for key quadruples that are to do parametrized encryption/decryption; and

H. repeating steps (E)-(G) of generating permutations for every key quadruple.

56. The method as claimed in claim 55, wherein the mappings to be encrypted are expressed using polynomials, further comprising the step of computing encryption and decryption mappings by interpolation, using at least a portion of R and S as interpolation data and using $a_i(x)$, once for each unique key quadruple that is to be generated.

57. The method as claimed in claim 55, further comprising the step of restricting a new set of key quadruples according to a pattern used for encryption.

58. The method as claimed in claim 55, further comprising the step of setting all key quadruples that are to do neither encryption nor decryption to the identity mapping.

59. A method of encrypting multivariate mappings with parametrized multivariate mappings, comprising the steps of:

A. determining appropriate mapping representation for the encryption;

B. symbolically substituting each group of variables $\vec{w}_{i-1}$ to be decrypted in a parametrized manner with $s_i(\vec{w}_{i-1},\vec{w}_{g_i-1})$;

C. symbolically substituting each group of variables $\vec{w}_{i-1}$ to be decrypted in a non-parametrized manner with $s_i(\vec{w}_{i-1})$;

D. symbolically composing each group of mapping components $\vec{v}_i$ to be encrypted in a parametrized manner with $r_i(\vec{v}_i, \vec{w}_{g,-1})$; and

E. symbolically composing each group of mapping components $\vec{v}_i$ to be encrypted in a non-parametrized manner with $r_i(\vec{v}_i)$.

60. A method of specifying an encryption pattern for parametrized encryption of a register machine, comprising the steps of:

A. defining a pattern of parametrized encryption;

B. setting all $c_i=d$ for a mapping doing computations of the register machine;

C. marking at least one register affected by a register transition mapping as plaintext registers;

D. marking a next instruction pointer and next storage pointer mappings as plaintext mappings;

E. marking key quadruples for the marked registers as not to be encrypted/decrypted;

F. marking the register transition mapping for non-parametric encryption;

G. marking the storage cell mapping q for parametric encryption; and

H. marking at least one "cell" in the storage space as plaintext "cells".

61. A method of key generation for parametrically encrypting a register machine, comprising the steps of:

A. determining a representation for keys;

B. defining, for a key quadruple that is to do parametrized encryption/decryption on storage cells, two temporary $N^{d+d}\times(d+1)$ arrays R and S;

C. defining for an i^{th} key quadruple that is to do non-parametrized encryption/decryption on register variables two temporary $N^d\times(d+1)$ arrays R and S;

D. defining a temporary polynomial f to translate from base N^d to base-N vectors with d components;

E. permuting a ring Z_{N^d}, and simultaneously translating the permutation and its inverse to a field Z_N^d;

F. setting the permutations to the identity mapping for the cells in the storage mapping marked as unencrypted;

G. repeating step (F) in all N^d times, except for cells in the storage mapping marked as unencrypted, while recording data, for the key quadruples that are to do parametrized encryption/decryption on the storage mapping;

09672050.121800

H. permuting the ring Z_{N^d}, and simultaneously translating the permutation and its inverse to the field Z_N^d; and

I. repeating the permuting step (H) for every key quadruple that is to encrypt a register transition mapping.

62. The method as claimed in claim 61, wherein mappings are represented using polynomials, further comprising the step of: computing i^{th} encryption and decryption mappings using in all $2d$ polynomial interpolations using R and S and a base-translation function f, for all i quadruples.

63. The method as claimed in claim 60, further comprising:
determining an appropriate representation for an encryption of the register machine; and
encryption the register machine.

64. The method as claimed in claim 61, further comprising the step of acquiring an set of key quadruples for use compositions.

65. The method as claimed in claim 10, wherein the steps (A)-(E) are performed by a computer.

66. The method as claimed in claim 13, wherein the steps (A)-(D) are performed by a computer.

67. The method as claimed in claim 14, wherein the steps (A)-(F) are performed by a computer.